CLAIMS

What is claimed is:

1. A compiler to compile a family of related functions, comprising:

a member recognizer configured to recognize a member function from said family of related functions;

a family start caller configured to make a family-start function call for said family of related functions; and

a member finish caller to make a member-finish function call for said member function.

2. The compiler of claim 1, further comprising:

an optimizer configured to optimize at least one of said family-start and member finish function calls.

3. The compiler of claim 2, wherein said optimizer is configured to optimize on at least one of intermediate language level, architecture specific level, and operating system specific level.

4. The compiler of claim 2, wherein said optimizer is configured to in-line expand at least one of said family-start and member-finish calls.

5. The compiler of claim 2, wherein said optimizer includes common subexpression elimination, code motion, and dead-code elimination.

6. The compiler of claim 1, wherein said family of related functions includes at least one of trigonometric, hyperbolic, and square root functions.

7. The compiler of claim 1, wherein said family of related functions is identified by use of a data store.

8. The compiler of claim 7, wherein said data store includes at least one of a look-up table, an ascii file, a binary file, and a database file.

9. The compiler of claim 7, wherein said data store is modifiable.

10. The compiler of claim 1, wherein one or both of said family start caller and said member finish caller are configured to make said family-start and member-finish function calls, respectively, in an intermediate language.

11. The compiler of claim 10, wherein said intermediate language is non-architecture specific and non-operating system specific.

12. The compiler of claim 1, wherein said member-finish function call makes use of a result returned from said family-start function call.

13. A method to compile a family of related functions, comprising:

recognizing a member function from said family of related functions;

making a family-start call for said family of related functions; and

making a member-finish call for said member function.

14. The method of claim 13, further comprising:

optimizing at least one of said family-start and member-finish function calls.

15. The method of claim 14 wherein in said optimizing step includes:

optimizing on at least one of intermediate language level and architecture specific level.

16. The method of claim 14 wherein said optimizing step includes:

in-line expanding at least one of said family-start and member-finish calls.

17. The method of claim 14, wherein said optimizing step includes common subexpression elimination, code motion, and dead-code elimination.

18. The method of claim 13 wherein said family of related functions includes at least one of trigonometric, hyperbolic, and square root functions.

19. The method of claim 13 wherein said recognizing step includes:

identifying said member function through a data store.

20. The method of claim 19 wherein said data store includes at least one of a look-up table, an ascii file, a binary file, or a database file.

21. The method of claim 19, further comprising:

modifying said data store.

22. The method of claim 13 wherein said family-start and member-finish function calls are made in an intermediate language.

23. The method of claim 22 wherein said intermediate language is non-architecture specific and non-operating system specific.

24. The method of claim 13 wherein said member-finish function call makes use of a result returned from said family-start function call.